Draft: Integration Coordination

A manual for Departmental Software Platforms and Services (DSPS)

Infrastructure Software Development and Architecture (ISDA), IS&T, MIT

Document Summary

This manual describes policies and recommendations regarding the management of application platforms by DSPS in ISDA. This will include the management of all assets required to reproduce and run server applications, such as source code, configuration, and procedures for release to production support.

This document is used to instruct work and audit service levels.

Table of Contents

Revision History

11/05/08	Steve Landry	Added Problem-Resolution

		and triage process. Added configuration-managment versioning.
10/30/08	Steve Landry	Updated configuration management information.
10/06/08	Steve Landry	Added information on support requests, email and ACL, RT queues.
09/07/08	Steve Landry	Changed installation procedures. Added Pipeline process information.
08/29/08	Steve Landry	Additions about managing acess-control lists
08/25/08	Steve Landry	Source-Control policies

Incident and Problem Management

This section tells you how to get application-management support from ISDA for problems with production services.

When in Doubt, Email isda-support@mit.edu

Customers external to IS&T should use **isda-support@mit.edu** to ask for help. Business-support personnel in ISDA receive these messages and track them in the Request Tracker (RT) queue **ISDA::Customer-Support**.

When a customer's problem requires technical resolution, the support ticket must be transferred or copied by ISDA customer support to the RT queue **ISDA::ProductionDefect**. The business-support person must also email **map-support@mit.edu**. This process should remain invisible to the external customer.

Support for Internal IS&T Technical Teams

IS&T staff who know they specifically require technical support for systems managed by DSPS can submit requests directly to **map-support@mit.edu**. DSPS tracks production problems in the RT queue **ISDA::ProductionDefect**.

Issue Response Playbook

DSPS team members should consult this playbook every time they participate in problem-resolution activity on behalf of ISDA.

Issue Definition

1. Our primary point of contact for support issues is the **map-support@mit.edu** mailing list. All system integrators in DSPS monitor this mailing list and the **ISDA::ProductionDefect** queue in RT.
    1. DSPS team members translate map-support emails into into RT tickets manually, in order to establish some clarity around the issue.
    2. If someone makes a support request through another channel, we email the issue to map-support and file it in RT or Jira.
    3. Automated responses from monitoring software come through to map-support@mit.edu, where they can be monitored and forwarded to the RT queue if necessary.
2. All team members in an *systems integration* role who are on the floor must meet to discuss the issue. A person is selected to lead the resolution.
    1. If possible, include non-DSPS staff in the discussion who are assigned to, or conversant with, the system in question.

3. The Tech Lead responds to the initiators of the message, alerting those parties that resolution is underway.
4. The Tech Lead must define the type of issue and then proceed accordingly.
    1. **Urgent Response:** A *system*, whether that is a whole server or a particular application, is unresponsive.
    2. **Bug Report:** An application is not doing the right thing in some particular case, but it is not generally broken; a system is not down.
    3. Both kinds of issues are priority #1. It is acceptable for all other responsibilities to be on hold until resolution (system down) or handoff (bug report).

Urgent Response (Problem Resolution)

1. For an unresponsive component, the team responsible for operating the system should attempt a restart of the server, application, or component evidencing the problem.[1]
    1. Someone familiar with the system must check to see if restart procedures occurred and if that temporarily resolved the problem.
2. Notification: preliminary problem description (and resolution, if applicable) sent to Recipient List:
    1. initiator of problem ticket, isda-pipeline@mit.edu, isda-integrators@mit.edu, map-support@mit.edu, and, if any end-user applications could have been affected, computing-help@mit.edu.
3. The Tech Lead must triage the process of determining causality of the issue with staff familiar with the system.
4. SCRUM: No resolution work should proceed until SCRUM is performed with available resources to discuss process and possible resolutions.
    1. Tech Lead is project manager for duration of issue resolution. Tech Lead is final arbiter for delegation of tasks, priorities, and timing.
5. Notification: If resolution is lengthy, Tech Lead will update Recipient List at least once per day of status of resolution.
6. Post Mortem: Tech Lead reviews response with DSPS manager. If emergency response offers the opportunity for improvement of process, Tech Lead calls a post-mortem with parties who participated in the resolution.

Bug Reports-Handoff (Change Management)

1. Tech Lead notifies a team leader or manager responsible for each tier of the system affected.
    1. Tech Lead collects information from managers on which mail lists to send notification of issue. This is the Recipient List for this issue. Note it in the ticket.

1 This playbook was designed in house and it is not backed up by research into industry-standard practices. Of note, current thinking on catching fragile parts of the system dictates that the operator should not automatically restart a service before determining causality.

    2. Tech Lead and managers determine staff members responsible for issue. This is the Team.
2. Tech Lead sends message to Recipient List notifying them of the issue.
3. Contact the Team and do preliminary troubleshooting to determine the nature of the issue and, therefore, the staff responsible to remedy the issue.
4. Transfer ticket information to system of record for the issue resolvers.
5. Notify the Recipient List of the transfer and the managers now responsible for the issue.

Change Management

Requesting a Planned Production Change

Please send deployment requests to **map-support@mit.edu**. DSPS will track production deployments in the RT queue **ISDA::ProductionDeploy**. These should have gone through the pipeline-planning process (see below).

Pipeline Planning

ISDA processes change requests through the ISDA Pipeline Planning process. ISDA management chooses the members of this group. All tasks related to new systems implementations, enhancement, or long-term issue resolution come through this group. You can reach this group at isda-pipeline@mit.edu.

RT tickets or full Daptiv project will be created to manage work internally.

Complete Contact List

isda-support@mit.edu: Primary list for customers outside of IS&T to request help and support.

map-support: List to submit requests for support directly to DSPS by internal IS&T teams. We manually create tickets for these requests into the appropriate ISDA support queue.

amit-auto-deploy@mit.edu: Email sent to AMIT via the ISDA::ProductionDeploy queue or tickets forwarded from ISDA::Customer-Support.

amit-auto-defect@mit.edu: Email sent to AMIT via the ISDA::ProductionDefect queue or tickets forwarded from ISDA::Customer-Support.

amit@mit.edu: Master member list for the Application Management and Integration Team, used for internal team email and access-controls. This list will always be an accurate representation of the system integrators currently in DSPS.

Request-Tracker Queues

ISDA::Customer-Support: ISDA's general customer-support queue.

ISDA::Admin: Work orders for any non-production application or system-integration work by AMIT.

ISDA::ProductionDefect: Specifically and only for tracking triage, troubleshooting, and resolution of problems with production enterprise services.

ISDA::ProductionDeploy: Specifically and only for tracking deployments to production enterprise services.

Release Management

You can request a release through **map-support@mit.edu**. See *Change Management*, above.

All systems are prepared as if they will be rolled out to data-center operations for production support. The reality is that DSPS supports some of its own production systems but formal release to Operations of all ISDA enterprise services is our ideal end state. Discipline in release management is guided by this principle.

Release Schedules

Productive Systems (staging or production): Release is variable. ISDA project coordinators negotiate a release window by coordinating with Pipeline, the project team, and the operational team responsible for the given system. This holds for systems defined as either "staging" or "production."

Non-Productive Systems (prototype, development, test): Systems are installed in such a way that implementation teams should have full access to system configurations they need to modify. This is not accomplished by full root or full admin access. If permissions are not sufficient, coordinate change requests through Pipeline.

Release Workflow

1. All services have four tiers: DEV, TEST, STAGE, PROD. Temporary prototype environments are available upon request.
2. Each tier of the service might have more than one machine in its topology to distribute load or for redundancy, but there are four "tiers" of release workflow regardless of service topology.
3. Development teams cannot access "stage" systems directly. A release must be bundled and tested for clean installation or patch onto staging by DSPS system integrators using standardized configurations whenever possible.
4. DSPS vets the installation for operational soundness. A successful installation or patch onto a staging environment results in a set of radmind loadsets used to release the service to production.
    1. This assures that a service can be rolled out to data-center operators and that those operators always have access to a last-known-good build of the service.

System Access Rules

Constraints on system access are designed to force a hand-off of installation and configuration, in order to evaluate installation procedures as operationally sound prior to release to data-center operators. We call this "vetting the run book."

- Only System Integrators in DSPS and System Operators in OIS have root access to any server. Only Only System Integrators in DSPS and System Operators in OIS have interactive shell access to "staging" or "production" systems.
- System integrators perform all configurations and deployments to any staging environment in order to derive a reproducible build process for rollout to production systems managed by OIS.
- Application administrators responsible for provisioning can have access to web-administration consoles in production environments for their products.
- All developers can have shell access to prototype, development, and test machines as users other than root.
- All developers can have access to any machine they support as the "logs" user, which will allow them read-only access to certain parts of the system.

We create **standard system-level users** on every machine: **db, logs, repos, and www**.

- Server applications run as these users, and not as root, for security reasons.
- Developers can use these accounts to access machines, since they cannot have root access.

The purpose of each account is as follows:

- "db" for MySQL or other database installations
- "logs" is to give developers read access to logs on staging or production servers.
- "repos" for Alfresco and other CMS installation
- "www" for Apache, Tomcat, and all other web-application server components

Application-Server Installation Standards

DSPS supports a menu of standard application-server components. We can redeploy these packages in any combination to provide a new project with a standardized platform upon which to build.

General	Java Application Servers	PHP Application Servers
VMware Red Hat Enterprise Linux 5.x Apache 2.2.x MySQL 5.x OpenSSL Shibboleth Development Environments Only: sFTP	Java Runtime Environment, Java Development Kit, versions 1.6.x SASHServer (SourceLabs Tomcat, based on 5.5.x) Tomcat 5.5.x	PHP 5.2.x PEAR

Source Control Management

We assume the use of Subversion as provided by IS&T via the enterprise installation at svn.mit.edu. This rules will update as the capabilities of that system change. DSPS only has

rules for tagging test releases and branching production releases. We do not plan to burden development teams with comprehensive source-control standard operating procedures.

Access Control, Separate ACL from Email

Our Subversion installation is integrated with Moira; access control lists are stored created in Moira. Subversion's ability to send email notifications is also stored in Moira.

1. Always use a separate Moira lists for repository access control versus email notification.
   a) Do not define the ACL as usable as an email list.
   b) Do not define the email list as usable for access control.
   c) The ACL can be the same as one used to control access to AFS lockers, remember that this list's membership should not spread past the project team or people to whom they report.
   d) This email list is separate and discrete from human-to-human mail lists for projects. Not everyone on a project needs to be alerted about source-control updates. This is optional.

Directory Structure

Conceptual	*[product]/[service]/[versioning]/[component]*
Pseudo	*[svn repos]/[top-level]/"trunk," "tags," or "branches"/[IDE Project or deployable unit]*

Product: The arch, meta-project, brand, or other executive concept. This should not be a team or organizational unit.

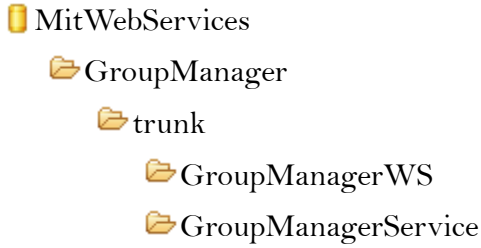
Service: That which is useful, runs, or operates as a whole, a conceptual piece of an overall product that you are likely to manage as a coherent project.

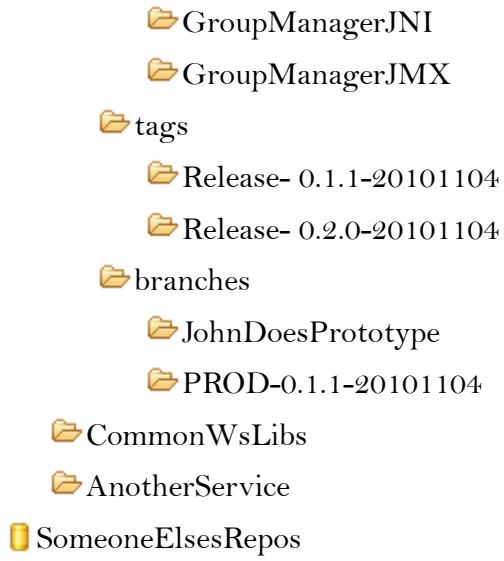
Versioning: The standard SVN directories of "trunk," "tags," or "branches" denoting the primary version-control function of SVN.

Component: A collection of code that compiles as a single deployable unit (a war file, tarball), that which functions well as an IDE project, or both.

Example

We'll use a fictionalized multicomponent web service as an example.





By this strategy, you end up with a GroupManager source tree where the web service, underlying libraries, and instrumentation code are all managed and versioned together even thought they have different deployment paths.

You alse end up with a top-level repository that is used to manage the whole "product," which can be vague but a meaningful categorization when projects change hands.

The content of the GroupManager sub-tree contains only that which is local to operating that particular service, but shared or dependent within that service. CommonWSLibs are tools shared and common to the overall MITWebServices product, like AnotherService.

Tagging for Release

Under "tags," Version Capture

"Release"-[version number]-yyyymmdd

We do not specify any particular scheme for version numbers. Products, projects, and technology are too varied.

When you are done unit testing and prepared to move a service to a test system, tag that service directory as a "Release," including your local version number and date.

Under "branches," Production Support

Once the *tagged* release is approved for production deployment, *branch* it to a production release in the form:

"PROD"-[version number]-yyyymmdd

Again, discrete version numbers are up to the project team. Leave them out of the name if you are not using them or tracking by date.

Configuration Management

Definitive Software Libraries (DSL)

DSPS tracks configurations of application-server components and configuration information about our systems in several ways.

amit-dsl AFS Locker

- The amit-dsl AFS locker contains DSPS archives of application installers, deployable components, reusable integration scripts, and other known good builds of application code.
- These archives correspond to the standard set of supported application server components listed in Release Management, above, plus supporting code for integration into the MIT data center environments.
- The locker also contains a protected archive visible only to DSPS where we keep licenses, system information, and other protected information about our software.
- The Athena File System is a service provided by OIS. It is replicated and backed up.

Radmind Server, Nebula

- The radmind tool is used to capture software installations with their localized installation properties and configuration information as loadsets.
- Reusable loadsets (repeatable builds) correspond to reference archives stored in the amit-dsl locker.
- Instance loadsets correspond to locally developed code or custom configurations of reusable components.
- The radmind server is a service provided by OIS. It is replicated across datacenters.

Run Books

Run-book documentation for any given system is stored in the ISDA wiki on wikis.mit.edu.

Installation Guidelines

- /usr/local is our preferred location for software installations.
    - Where we use standard packages (RPM), installed software stays in its default location.
    - Where a particular product encourages a different installation location, we will go by the project's recommendation unless experience teaches us otherwise.

- /usr/local/src is where we keep packages that were installed locally on the system.
- /usr/local/etc is where we keep properties files and other configuration information for locally-built applications.
- RPM based installations must use the package that comes with the operating system in question so as not to corrupt the system's installation database.
    - When our needs dictate a package newer than the one provided by the OS, we will produce our own build managed via our own DSL procedures.

Versioning Radmind Loadsets

The easiest way to explain this is with an example. Here is how we versioned Apache:

was-apache-bin-rhel5-current.T (symlink to latest stable build)

was-apache-bin-rhel5-001.001.T (concatenated node builds)

was-apache-bin-rhel5-001.001 (corresponding directory to *.T file)

was-apache-bin-rhel5-001.001-node1.T

was-apache-bin-rhel5-001.001-node1 (corresponding directory to *.T file)

was-apache-bin-rhel5-001.001-node2.T

was-apache-bin-rhel5-001.001-node2 (corresponding directory to *.T file)

was-apache-cfg-dev-001.001.T

was-apache-cfg-dev-001.001 (corresponding directory to *.T file)

was-apache-cfg-stable-001.001.T

was-apache-cfg-stable-001.001 (corresponding directory to *.T file)

was-apache-cfg-stale-001.001.T

was-apache-cfg-stale-001.001 (corresponding directory to *.T file)

was-apache-neg-rhel5-001.001.T

was-apache-neg-rhel5-001.001 (corresponding directory to *.T file)

Component Family	was = web application server	
Component	apache, jdk, php, sashserver	
Loadset Type	bin, cfg, neg	Binary, configuration, or negative loadset.
OS Compatibility	rhel4, rhel5	
Versions for "cfg" loadsets	dev, stable, stale	
Versions for "bin" loadsets	xxx.xxx, current (symlink only)	The first set of numbers iterates for major reconfigs,

		the second for minor changes that can be done without recapturing the loadsets from a prototype rebuild.
Versions for nodes	node	A node is a point on the directory tree. Node build numbers correspond to what the build number will be for the concatenated loadset of all node builds.

Service Continuity and Availability

[TBD: This section will include standard system topologies, backup and replication patterns, and other concerns, most of which are in practice in ISDA today. ed.]

Bibliography

Behr, Kevin, Gene Kim, and George Spafford. The Visible Ops Handbook, Implementing ITIL in 4 Practical and Auditable Steps. Revised 1st Edition. Eugene, OR: ITPI, 2007.

Disaboto, Michael. ITIL Service Management Processes, Third Time's the Charm. Midvale, UT: Burton Group, 2007.